Introduction to Go

Richard Liu

17 January 2017

ECE 458

History and Overview

- Go was developed in 2009 by engineers at Google due to dissatisfaction with other systems languages (C/C++)
- Go is statically typed and compiled, and supports garbage collection, memory safety, multiprogramming, networking
  - Optimizes compilation speed with better dependency analysis (no header files)
- Easy to read
- Compiled with either gc or gccgo

Pros and Cons

- Pros
  - Modern systems language with garbage collection
  - Optimized for multicore machines using concurrency
  - Very scalable
  - Fast compilation
  - Easy to read and write
- Cons
  - No type inheritance, generics, pointer arithmetic, overloading, exceptions, assertions

Syntax

- Very readable
- Functions, variables, control flow all pretty similar to Java
- For pointers, syntax is like C, but there is no pointer arithmetic

```
1  package main
2  import (
3      "fmt"
4      "math"
5  )
6
7  func pow(x, n, lim float64) float64 {
8      if v := math.Pow(x, n); v < lim {
9          return v
10     } else {
11         fmt.Printf("%g >= %g\n", v, lim)
12     }
13     // can't use v here, though
14     return lim
15 }
16
17 func sum() {
18     sum := 1
19     for sum < 1000 {
20         sum += sum
21     }
22     fmt.Println(sum)
23 }
24
25 func main() {
26     fmt.Println(
27         pow(3, 2, 10),
28         pow(3, 3, 20),
29     )
30 }
31
```

Syntax (cont'd)

- Like C, Go supports structs
  - Structs have no classes, but you can define methods on types
- Like in Python, you can "slice" arrays

```
package main

import "fmt"

type Vertex struct {
    X, Y int
}

var (
    v1 = Vertex{1, 2}  // has type Vertex
    v2 = Vertex{X: 1}  // Y:0 is implicit
    v3 = Vertex{}      // X:0 and Y:0
    p  = &Vertex{1, 2} // has type *Vertex
)

func main() {
    fmt.Println(v1, p, v2, v3)
}

```

```
func (v Vertex) Abs() float64 { // use v.Abs() to call
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}
```

```
package main

import "fmt"

func main() {
    primes := [6]int{2, 3, 5, 7, 11, 13}

    var s []int = primes[1:4]
    fmt.Println(s) // [3 5 7]
}

```

Syntax (cont'd)

- Go supports interfaces
  - Implement interface by implementing methods. No "declaration" required
- Concurrency management via "goroutines"
  - Use "go" keyword to start a new lightweight thread
  - Use "channels" to send and receive data

```
package main

import "fmt"

type I interface {
    M()
}

type T struct {
    S string
}

// This method means type T implements the interface I,
// but we don't need to explicitly declare that it does so.
func (t T) M() {
    fmt.Println(t.S)
}

func main() {
    var i I = T{"hello"}
    i.M()
}

```

```
package main

import (
    "fmt"
    "time"
)

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
    // world
    // hello
    // world
    // hello
    // hello
    // world
    // world
    // hello
    // hello
    // world
}

```

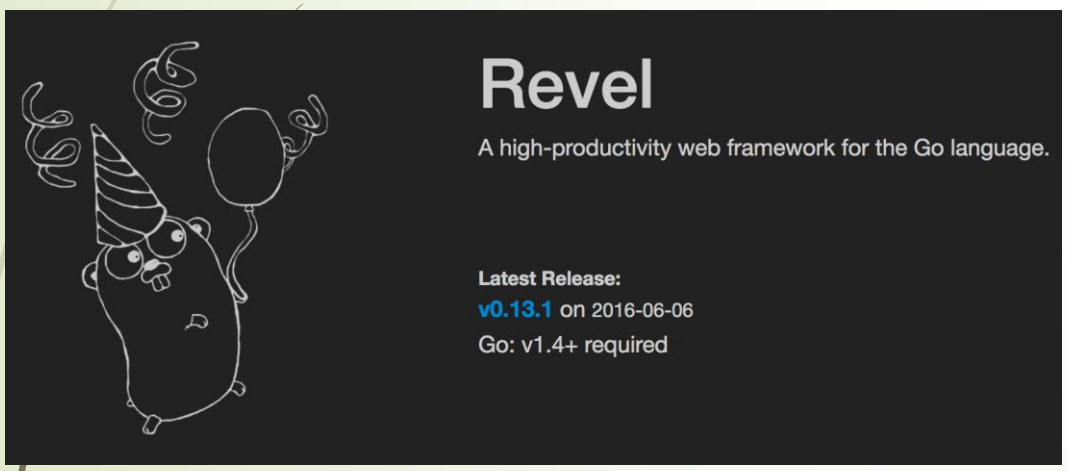
Go Command

- https://golang.org/cmd/go/
- Supports package management
- Go help [command] to help
- Useful packages:
  - Crypto, sql, dwarf (debugging), csv, json, xml, html, io, net, os, text

```
build       compile packages and dependencies
clean       remove object files
doc         show documentation for package or symbol
env         print Go environment information
fix         run go tool fix on packages
fmt         run gofmt on package sources
generate    generate Go files by processing source
get         download and install packages and dependencies
install     compile and install packages and dependencies
list        list packages
run         compile and run Go program
test        test packages
tool        run specified go tool
version     print Go version
vet         run go tool vet on packages
```

Web Framework

- Revel (https://revel.github.io/)



- Goji (https://goji.io/)

```go
package main

import (
        "fmt"
        "net/http"

        "goji.io"
        "goji.io/pat"
)

func hello(w http.ResponseWriter, r *http.Request) {
        name := pat.Param(r, "name")
        fmt.Fprintf(w, "Hello, %s!", name)
}

func main() {
        mux := goji.NewMux()
        mux.HandleFunc(pat.Get("/hello/:name"), hello)

        http.ListenAndServe("localhost:8000", mux)
}
```

Helpful Links

- Installation (https://golang.org/doc/install)
  - Download at https://golang.org/dl
- Compilers:
  - gc, gccgo
- Wiki: https://github.com/golang/go/wiki
- Tutorials
  - https://**tour**.golang.org/
  - https://golang.org/doc/code.html
- Examples
  - https://golang.org/doc/effective_go.html
- Go command
  - https://golang.org/cmd/go/
- Revel (https://revel.github.io/)
  - Quick Start:
    - go get -u github.com/revel/cmd/revel
    - revel new github.com/myaccount/my-app
    - revel run github.com/myaccount/my-app
    - open https://localhost:9000 in browser